Name: Manu Bhargava Reddy Nannuri

andrew ID: mnannuri@andrew.cmu.edu

Homework 1

1 Statement of Assurance

I certify that all of the material that I submitted is my original work and that it is done only by me.

2 Structured query set

2.1 Summary of query structuring strategies

Below are some of the query structuring strategies that I have used to obtain a balanced trade off between precision and recall.

* For common phrases like “New York”, “Cheap Interner” etc. its better to use near operator with a little n. This would improve the precision while not decreasing the recall value drastically.

* Certain common words are betters to be searched both among the title as well as body.

* Extremely rare words which often are abbreviations or acronyms are better of searching in the url or inlinks only. This is true because, these acronyms or abbreviations usually appear in their full form in the body and hence are trickier to find.

*

Query 10:#NEAR/3(cheap.title internet.title)

This is a common phrase and therefore a near operator is best suited to find the occurrence of this phrase.

Query 12:#AND(djs.title djs.inlink djs.body)

djs is a very common word and since there might be many web pages which happen to have djs listed on them. Searching for the word in the title, inlink and body would be a good choice.

Query 26:#AND(lower #NEAR/2(heart rate))

Heart rate is a popular phrase and hence applying the near operator would fetch documents which are relevant to heartrate only. The and operator just fetched those document which have lower and heart rate listed in the same page. I also tried having a nested near including lower and the inverted list fetched by the other near operator. It didnt fetch many relevant docs as our near operator only returns the position of the last word and hence I couldnt estimate a suitable value for n in the outer near operator.

Query 29:#AND(#SYN(#NEAR/3(ps 2) PS2 PLAYSTATION) games)

ps 2 PS2 and playstation are interchangeably used in a wide variety of settings. Hence the syn operator would capture all those occurrences and fetch us those results which have games occuring along with the ps2 argument.

Query 33:#NEAR/4(elliptical trainer)

Elliptical trainer is a commonly occuring phrase and hence searching for it using the near operator would fetch us the maximum number of results. I chose the value of n as 4 because I suspected elliptical might be succeeded by its other popular term “cross”.

Query 52:#OR(avp.url)

avp looks like an acronym. Hence there is a higher probability that it might be present in a url or inlink text. Hence we formed the query using only avp.url

Query 71:#OR(#NEAR/4(living.title india.title) #AND(living india))

Many of the pages which talk about living in india might have “living india“ in the title itself. I didnt want to loose out on general pages or forum pages which talk about living in india. Hence, I also included the AND operator on “living” and “india”

Query 102:#NEAR/3(fickle creek farm)

Fickle creek farm is a name and hence these tokens always occur together. Hence the usage of the near operator might be apt to retrieve the relevant document.

Query 149:#AND(uplift yellowstone national park #NEAR/3(yellowstone park))

This information need talks about upliftment at Yellowstone National Park. A relevant result might have all the key words and the entire phrase “yellowstone park”. This is the rationale behind structuring the query this way.

Query 190:#AND(#NEAR/2(brooks.title brothers.title) clearance)

Similar to the previous query. Brooks brothers is a popular store and brooks brothers is a phrase and has a high chance of occurring in the title itself. so relevant pages would have brooks brothers in a title and clearance else where(most likely the body. It could also be the url or title).

3 Experimental results

3.1 Unranked Boolean

	BOW #OR	**BOW #AND**	**Structured**

P@10	0.0100	0.0400	0.1800
P@20	0.0050	0.0200	0.2350
P@30	0.0033	0.0433	0.2100
MAP	0.0010	0.0142	0.0912
Running Time	12:30	00:18	00:15

3.2 Ranked Boolean

	BOW #OR	**BOW #AND**	**Structured**
P@10	0.1500	0.2500	0.2800
P@20	0.1800	0.2600	0.3250
P@30	0.1667	0.2767	0.3300
MAP	0.0566	0.0980	0.1500
Running Time	15:15	00:17	00:15

***All the running times are measured on an old dual core laptop running ubuntu 14.04(4 gb ram). I believe this might be the reason for the exceptionally high running time. I have tested this on another machine and the running times were remarkably less. Kindly take this into consideration.

4 Analysis of results

As part of the assignment, I implemented the unranked boolean retrieval algorithms and the ranked boolean retrieval algorithm. The ranked boolean retrieval algorithms is better over its unranked counterpart in many ways. It makes the results predictable and easy to explain. This becomes even more evident when we return a limited number of results.

The OR operator is basically a union of the inverted lists of all the terms present in the query. This would give us every page which has at least one of the terms listed in it. This results in a higher recall and lower precision. The ranking scheme that we used to order the results is a max operation on the score of the individual document. This would rank all those documents which have many occurrences of a single term as highly relevant document which might not be true in all the cases. A better way to rank our results is using an average operator. One more observation I had wrt OR operator is that it is pretty much the same as the syn operator in the unranked case but it would differ in the ranked scenario.

The AND operator gives the intersection of the inverted lists of all the terms in the queries. This means that each and every term must be present in all the documents. Applying the AND operator would result in a higher precision and low recall. This is evident from the comparison of MAP values in the BOW #OR and BOW #AND columns. This usually returns the most relevant documents and is preferably used in the commercial search engines as the average search user values high precision at the cost of recall. The

AND operator takes the minimum amount of time because we sort the lists in increasing order of their document frequency and terminate when we are done with the smallest list.

The NEAR operator returns those document which have the matching terms occurring within a specified distance. This implementation of a near operator also places a constraint on the term order. The near operator resembles the AND operator by ensuring that all the term occur in the document and also promotes the relevance of the query by ensuring that the terms occur within a specified distance. This will boost the number of relevant document retrieved for an information need. The most common example where the near operator is most used is while searching for phrase words like “new york” or “california university” etc. The near operator would most likely give those results which are relevant but the and operator might just result all those documents which just have all the terms present in them. The near operator would obviously take more time compared to the “and” operator as it checks for positions of the terms in the documents as well.

The use of structured queries is a good way to achieve a satisfactory tradeoff between precision and recall. As the results show, the structured queries for the same information needs report a higher MAP score and report better precision numbers at various recall percentages. These structured queries use a combination of various basic operators and hence might usually result in higher memory usage and running times than the AND operator. This is because there are many more operations that need to be performed and also there are many more intermediate result sets that need to be stored in memory.